
INDEX

abnormal termination, SR, 253
Abrahams, P. W., 179, 353
accept statement, Ada, 204
 Cell, 264
accept, select alternatives, 206
access rights, Concurrent Pascal, 184
access types, Ada, 212
Ackerman, W. B., 127–128, 347
acquaintances, Actors, 146
Act-1, 149–150
Act-2, 149
action's priority, Hearsay-II, 304
Actor semantics, 150
actor, Data Flow, 114
Actors, xviii, 145–153, 156, 230, 233, 245,
 310, 325–345
acquaintances, 146
complaint box, 146
laws for, 148
messages, 146
model, 145
pattern-matched invocation, 149
primitive, 146
scripts, 146
serialized, 146–147
target, 146
unserialized, 146
Ada, xvii–xviii, 201–227, 230, 245, 248, 261,
 263–264, 268, 271, 325–345
accept statement, 204
access types, 212
conditional entry call, 214
critical region, 205
delay, 206
dynamic exception handling, 225
encapsulation, 203
entry, 203
interrupts, 215
modules, 204
new, 212
packages, 224
processes, 202
raise statement, 225
rendezvous, 205
select statement, 206
select/entry call, 214–215
synchronized communication, 202, 226
task priority, 224–225
task termination, 203
task types, 212
tasks, 203, 232
termination, 225–226
timed entry call, 215
when statement, 206
Adams, D. A., 106, 128, 347
Adiba, M. E., 288, 299, 347
Agerwala, T., 105, 127–128, 347
Aho, A. V., 15, 347
Ahuja, V., 43, 347
airline reservation system, solution in Data
 Flow, 124–125
Alagic, S., 18, 23, 347
Algol 60, 149, 158
algorithmic selection, solution in IAP,
 171–172
Allen, J., 160, 177, 347

- Allison, D.C.S., 8, 15, 351
 Alphard, 19
 amb, 81, 165, 336
 Ames, W. F., 178, 352
 analysis of algorithms, 12–14, 326
 Andrews, G. R., 246–247, 269, 274, 325, 346–347
 antifairness, 51, 333
 applicative languages, 17
 applicative-order evaluation, 158
 Apt, K. R., 130, 141, 144, 347
 Arbib, M. A., 18, 23, 347
 architecture, xv
 Argus, 279, 290–298, 325–345
 atomic actions, 290
 concurrency, 294
 exception handlers, 294
 guardians, 290
 handlers, 290
 processes, 294
 read locks, 292
 recovery, 290
 stable storage, 290
 subactions, 291
 topactions, 291
 two-phase commit, 290
 versions, 292
 volatile storage, 290
 write locks, 292
 array of processes, CSP, 135
 artificial intelligence, 149, 310
 Arvind, 106, 127–128, 346–347
 Ashcroft, E. A., 162, 177, 348
 asymmetric communication control, 50
 asynchronous communication, 327
 PLITS, 230
 asynchronous message transmission, 49
 asynchronous sends, SR, 246
 Atkinson, R. R., 24, 300, 353
 atomic action, 36
 atomic actions, 200
 Argus, 290
 failure, 52
 heuristic mechanisms, 52
 atomicity, distributed databases, 280–286
 attacking rovers game, 318
 Attardi, G., 154, 351
 Augusta, A., 202
 automata theory, 3–8
await statement, Cell, 264
 axiomatic, semantics, 18
 Babbage, C., 202
 background routine, guardians, 293
 Backus, J., 17, 23, 348
 Baker, H., 148, 154, 351
 bandwidth, communication, xiii
 bank account, solution in Actors, 152–153
 batch/stream computers, xi
 Bayer, R., 299, 351
 Bell, G., 41, 43, 355
 Berger, P., 106, 128, 348
 Bernstein, A. J., 130, 143–144, 215, 228, 348
 Bernstein, P. A., 280, 286, 299, 348
 bid specification, Contract Nets, 307
 bidding, Contract Nets, 307
 bidirectional delayed information flow, 50, 329
 bidirectional information flow, 50, 329
 bidirectional simultaneous information flow, 50, 329
 binary semaphore, 35
 solution in Concurrent Processes, 92
 solution in Distributed Processes, 194
 solution in Exchange Functions, 78
 Birtwistle, G. M., 19, 23, 348
 blackboard, Hearsay-II, 304, 310
 Blasgen, M., 299, 351
 blocked processes, 27, 49
 Bobrow, D. G., 322, 356
 Boggs, D. R., 85, 353
 boolean actors, Data Flow, 116
 bounded buffers, 50, 329
 Braffort, P., 179, 353
 Brinch Hansen, P., 183, 188, 191, 198–199, 348
 broadcast, blackboard, 326
 communication connection, 51
 broadcasting, Contract Nets, 307
 Brodie, M. L., 321, 351
 Brownbridge, D. R., 127, 129, 355
 Buckley, G., 143–144, 348
 buffers, bounded, 50, 329
 unbounded, 50, 329
 Burns, J. E., 66, 72, 348
by, Cell, 267
 SR, 248
 call-by-name, 149, 158
 call-by-need, 159
 in Exchange Functions, 82
 lazy evaluator, 159
 networks of parallel processes, 159

- suspending cons, 159
- call-by-reference, 204
- call-by-value-result, 204
- card reader, interrupt, 101
 - solution in Concurrent Processes, 95–103
- Carlson, W. E., 201, 228, 348
- Cell, 245–246, 261–267, 325–345
- Cell extension, 271–273
- Cell, **accept** statement, 264
 - await** statement, 264
 - by**, 267
 - partial order in **select** statement, 267
 - partial-order priority, 273
 - process termination, 262, 268
 - processes, 261
 - scheduling, 263–265
 - select** statement, 264
- Charniak, E., 322, 355
- Christensen, M., 178, 352
- Church, A., 7–8, 10, 15, 179, 348
- Church-Rosser theorem, 10
- Church's thesis, 7
- Clark, K. L., 176, 178, 348
- clash, 93
- classes, Concurrent Pascal, 184, 187
- clause, 93
- Clinger, W., 150, 154, 348
- Clossman, G., 228
- closures, 19, 149–150
- CLU, 19, 24, 290, 300
- Cobol, xv
- coherent problem solving, xiii
- Collins, A., 322, 356
- combinatorial implosion, 311–316
- common procedures, Distributed Processes, 192
- Communicating Distributed Processes, dining philosophers, 200
- Communicating Sequential Processes, *see* CSP
- communication, xiii–xv
 - communication connection, broadcast, 51
 - entry, 51
 - name, 51
 - port, 51
 - communication control, asymmetric, 50
 - symmetric, 50
 - communication delay, solution in Shared Variables, 67–68
 - communication matches, Exchange Functions, 76
- communication, bandwidth, xiii
- commutativity, Scientific Community Metaphor, 310
- complaint box, Actors, 146
- complement, 93
- computer networks, xiii
- Comte, D., 106, 128, 348
- concurrency, xv
- concurrency control, distributed databases, 279, 283
- concurrency, Argus, 294
- concurrent computation, xiv
- concurrent languages, xv
- Concurrent Pascal, 36, 184–191, 196, 200, 325–345
 - access rights, 184
 - classes, 184, 187
 - monitors, 184
 - peripheral devices, 187
 - processes, 184
 - storage allocation, 188
- Concurrent Processes, xvii, 86–103, 325–345
 - continuations, 87
 - correspondence with Shared Variables, 90
 - covert communication, 88–89
 - label renaming, 90
 - mathematics of composition, 89–90
 - net, 88
 - ports, 87
 - processes, 87
 - renewal, 89
 - restricting label visibility, 90
 - sort, 88
 - conditional entry call, Ada, 214
 - conditionals, solution in Data Flow, 117–119
 - conditions, Petri Nets, 110
 - conjunctive normal form, 93
 - connection, full-duplex, 50
 - half-duplex, 50
 - simplex, 50
 - cons, 326, 336
 - suspending, 159–161
 - constant actors, Data Flow, 117
 - continuation-based architectures, 317
 - continuations, 146, 149, 242
 - in Concurrent Processes, 87
 - continue**, monitors, 185
 - continuous display, solution in Shared Variables, 70–71
 - contract award, Contract Nets, 307

- contract manager, Contract Nets, 307
- Contract Net Protocol, 316
- Contract Nets, 303, 307–309, 325–345
 - bid specification, 307
 - bidding, 307
 - broadcasting, 307
 - contract award, 307
 - contract manager, 307
 - contractor, 307
 - contracts, 307
 - directed contract, 307
 - eligibility specification, 307
 - limited broadcast, 307
 - monitor node, 308
 - point-to-point announcement, 307
 - problem solving protocols, 307
 - processes, 308
 - processing node, 308
 - request-response sequence, 307
 - sensing node, 308
 - task description, 307
 - task report, 307
- contractor, Contract Nets, 307
- contracts, Contract Nets, 307
- control links, Data Flow, 115
- control system, elevator, 215
- Conway, L., 41, 43, 353
- Conway, M. E., 19, 21, 23, 348
- Cook, S. A., 15, 348
- coordinated computing, xi–xii, xiv, xvi
- Corkill, D. D., 302, 317–318, 320–321, 349, 353
- coroutines, 19
- correctness proofs, xviii
- Courtois, P. J., 36, 40, 349
- covering set problem, solution in Ether, 311–316
- covert communication, Concurrent Processes, 88–89
- creation routines, guardians, 293
- Cremers, A., 66, 72, 349
- critical region, 27
 - Ada, 205
 - rendezvous, 205
- CSP, xvii–xviii, 130–143, 194, 206, 227, 232, 245, 248, 325–345
 - array of processes, 135
 - failure, 132
 - guarded input command, 134
 - guarded statements, 134
 - input statements, 131
 - lack of output guards, 143
- livelock, 143
- output guards, 215
- output statements, 131
- processes, 131, 133
- statements, 132
- cull**, solution in IAP, 174–176
- cycle**, Distributed Processes guarded commands, 193
- dagger, xvii
- Dahl, O.-J., 23, 178, 348–349
- data abstraction, 19, 145
- Data Flow, 114–127, 156–157, 325–345
 - actor, 114
 - boolean actors, 116
 - constant actors, 117
 - control links, 115
 - data links, 115
 - deciders, 116
 - elementary actors, 115–117
 - firing rules, 115
 - gates, 117
 - indeterminate-merge actor, 122
 - link, 115
 - operators, 116
 - program, 115
 - simple-merge actors, 117
 - data links, Data Flow, 115
 - data recursion, 162–165
 - database snapshots, 288
 - database systems, 279
 - Date, C. J., 279, 299, 349
 - Davies, C. T., 291, 299, 349
 - Davis, A. L., 106, 128, 349
 - Davis, R., 307, 310, 320, 322, 349, 355
 - de Jong, P., 315, 321, 351
 - de Roever, W. P., 141, 144, 347
 - deadlock, 27
 - deadlock detection, 284
 - waiting-for relation, 285
 - deciders, Data Flow, 116
 - declarative-procedural controversy, 322
 - Dekker, T., 33–34, 39–40, 60, 62
 - delay, Ada, 206
 - delay**, monitors, 185
 - select** alternatives, 206
 - delayed evaluation, 158
 - delaying exchange, Exchange Functions, 82–84
 - demand driven systems, 159
 - demons, 310
 - demultiplexer, solution in Data Flow, 120–121

- Dennis, J. B., 36, 40, 106, 114, 119, 122, 128, 346–347, 349
denotational semantics, 18
die, transaction, 286
Dijkstra, E. W., 20, 23, 28, 35, 37, 40, 60, 81, 163, 170, 178, 349
dining philosophers, 36
 in Communicating Distributed Processes, 200
 solution in Ada, 212–214
 solution in Argus, 294–298
 solution in Concurrent Pascal, 188–189
 solution in CSP, 137–138
 solution in Distributed Processes, 194–197
 solution in Petri Nets, 111–112
directed contract, Contract Nets, 307
Dislang, 200
distributed computing, 41
distributed databases, 326
 atomicity, 280–286
 concurrency control, 279, 283
 failure and recovery mechanisms, 279
 motivations, 280
 replication, 279–280, 286–290
 serializability, 282
 transactions, 280–286
 transparency, 280
distributed demand-driven bus system
 problem, 318
Distributed Hearsay, 303–306
Distributed Hearsay-II, 325–345
distributed languages, xv
distributed operating systems, xiii
distributed problem solving, 326
Distributed Processes, 191–197, 202, 230, 245, 248, 290, 294, 325–345
Distributed Processes guarded commands,
 cycle, 193
 do, 193
 if, 193
 when, 193
Distributed Processes, common
 procedures, 192
 initial statement, 192
 local storage, 192
 name, 192
distributed systems, xiii
do, Distributed Processes guarded
 commands, 193
Donzeau-Gouge, V., 202, 228, 349
Dwyer, R. A., xviii, 349
Dybvig, R. K., 349
dynamic exception handling, Ada, 225
dynamic process creation, 48
early computers, xi
economic, motivations, xv
Edison, 198, 200
Edwards, D. J., 179, 353
eight queens, solution in PLITS, 240–243
Elcock, E. W., 320, 349
election algorithms, solution in Shared
 Variables, 68–70
elementary actors, Data Flow, 115–117
elevator controller, 103
 solution in Ada, 215–224
eligibility specification, Contract Nets, 307
else, **select** alternatives, 207
embedded systems, 74, 201, 215, 326
encapsulation, Ada, 203
Enea, H. J., 106, 129, 355
entry, Ada, 203
 communication connection, 51
Erman, L. D., 302, 304–305, 320–321,
 350, 353
Eswaran, K. P., 283, 299, 350
Ether, 149, 309–316, 325–345
 sprites, 310
Ethernet, 77, 142
events, Petri Nets, 110
exception handlers, Argus, 294
 failure, 52
exception handling, 201
 failure, 333
Exchange Functions, xviii, 74–77, 87, 227,
 271, 325–345
 call-by-need, 82
 communication matches, 76
 delaying exchange, 82–84
 extended semantics of **XR**, 83
 instantaneous primitive, 76
 join-by-need, 82
 real-time clock, 76
 real-time sensing, 76
 solution in Cell, 265
 successor functions, 74, 77
 X, 75
 XM, 76
 XR, 75
exclusive (write, update), lock, 283
exercises, xvii
explicit process systems, 326
explicit processes, 48

- exponentiation, solution in Data Flow, 119–120
 exprs, in Lisp, 158
 extended semantics of $\times R$, Exchange Functions, 83
- factorial, solution in Actors, 151–152
 failure, xv
 failure and recovery mechanisms, distributed databases, 279
 failure, atomic actions, 52
 CSP, 132
 exception handlers, 52
 exception handling, 333
 frons, 333
 functional accuracy, 52
 redundancy, 52
 time-outs, 52, 333
 fairness, antifair, 51, 333
 strong fairness, 51, 333
 weak fairness, 51, 333
 Feigenbaum, E. A., 322, 354
 Feldman, J. A., 230–231, 233, 244, 338, 346, 350
 Fennell, R. D., 305, 320, 350
 Fibonacci numbers, solution in CSP, 133
 solution in PLITS, 233–235
 filling **split**, 337
 Filman, R. E., 346, 350
 finite-state automata, 6
 firing of transitions, Petri Nets, 109
 firing rules, Data Flow, 115
 Fischer, M. J., 57–58, 60–61, 64, 71–73, 90, 340, 348, 353
 Fisher, D. A., 128, 201, 228, 349–350
 Fitzwater, D. R., 74–77, 84–85, 350, 356
 focus of control, Hearsay-II, 304
 FOL, 322
 forcing, suspending **cons**, 160
fork, 21
 formal semantics, xviii
 Forsythe, A. I., 16, 24, 354
 Fosseen, J. B., 106, 129, 350
 Fox, M. S., 320–321, 350
 Francez, N., 141, 144, 347, 350
 Friedman, D. P., 159–160, 163–164, 166, 168–169, 178, 346, 350
 frons, 166–168, 326, 336
 failure, 333
 full-duplex connection, 50
 funargs, 149
 function graph language (FGL), 179
- functional accuracy, failure, 52
 functional recursion, 161
 Functionally Accurate Cooperative Systems, 302–303
 fusion, 93
- Galbraith, J., 321, 350
 Garman, J. R., 333, 346, 350
 gates, Data Flow, 117
 Gauss, C. F., 240
 Genuys, F., 40, 349
 Gilchrist, B., 154, 178, 346–347, 352
 Ginder, J. R., 356
 Goldberg, A., 19, 23, 350
 good sequences, solution in IAP, 170
 Goodman, N., 280, 286, 299, 348
 Gordon, M.J.C., 18, 23, 129, 351
 Gostelow, K. P., 106, 128, 346–347
 Graham, R. M., 299, 351
 Gray, J. N., 286, 289, 291, 299, 350–351
 Gregory, S., 176, 178, 348
 Greif, I., 150, 154, 351
 Gries, D., 18, 24, 141, 144, 351, 353
 guarded commands, 20
 SR, 248
 guarded exchange functions, 81–84
 guarded input command, CSP, 134
 guarded statements, CSP, 134
 guarded, **select** alternatives, 206
 guardians, 293–294
 Argus, 290
 background routine, 293
 creation routines, 293
 handlers, 293
 names, 293
 recovery routine, 293
 stable storage, 293
 volatile storage, 293
 Gurd, J., 106, 129, 356
- half-duplex connection, 50
 halting problem, 7
 Petri Nets, 114
 Hamming, R. W., 163–164, 178
 Hamming sequence, solution in IAP, 163
 Hanson, A. R., 304, 321, 351
 hardware, xiv
 hardware semantics, 326
 Hart, T. P., 179, 353
 Hayes-Roth, F., 320, 322, 350, 354
 Heapsort, 138
 Hearsay, 302

- Hearsay-II, 309–310, 316
 action's priority, 304
 blackboard, 304, 310
 focus of control, 304
 knowledge sources, 304
 scheduler, 304
 speech understanding, 304
 Henderson, P., 159, 173, 178, 351
 heterarchical control, 303
 heuristic mechanisms, atomic actions, 52
 negotiation-based control, 52
 pattern-directed invocation, 52
 heuristic systems, 326
 heuristics, xiv–xv
 Hewitt, C. E., 145–146, 148–154, 310, 315–316, 321–322, 351–352
 Heymans, F., 36, 40, 349
 Hibbard, T., 66, 72, 349
 hierarchical control, 303
 Hifdi, N., 128, 348
 Hirschberg, D., 179, 353
 Hirschberg, D. S., 70, 72, 351
 historical perspective, xi
 Hoare, C.A.R., 8, 15, 36, 40, 130–133, 135, 137, 141, 143–144, 178, 185, 199, 227–228, 349, 351–352
 Holt, A. W., 105–106, 129, 352
 Hopcroft, J. E., 15, 347, 352
 Hopkins, R. P., 127, 129, 355
 Horn, B.K.P., 179, 356
- IAP, 156–176, 325–345
 indeterminacy, 165–170
 send-and-forget, 169
 side effects, 298
 split, 174
 sting, 169
 test-and-set, 170
 Ichbiah, J., 202
if, Distributed Processes guarded commands, 193
 imperative languages, 17
in statement, SR, 248
 incremental log, 288
 indeterminacy, IAP, 165–170
 Indeterminate Applicative Programming,
 see IAP
 indeterminate-merge actor, Data Flow, 122
 infinite objects, 157–159
 infinite sequences, 157
 information flow, bidirectional, 50, 329
 bidirectional delayed, 50, 329
 bidirectional simultaneous, 50, 329
 unidirectional, 50, 329
 Ingerman, P., 19, 24, 149, 154, 352
 initial operation, monitors, 185
 initial statement, Distributed Processes, 192
 input statements, CSP, 131
 instantaneous primitive, Exchange Functions, 76
 intellectual, motivations, xv
 interrupt, card reader, 101
 interrupts, Ada, 215
 Jackson, P., 72, 348
 Johnson, S. D., 164, 170, 172, 178, 352
join, 21
 join-by-need, in Exchange Functions, 82
 K combinator, 91
 Kahn, G., 154, 157, 159, 178, 228, 349–352
 Karp, R. M., 106, 129, 352
 Keller, R. M., 173, 179, 352
 Kent, E., 320–321, 352
 Kessels, J.L.W., 264, 274, 352
 Kieburtz, R. B., 70, 72, 130, 142–144, 227–228, 352
 King, J., 310, 320, 349
 knowledge sources, 310
 Hearsay-II, 304
 Knuth, D. E., 14–15, 352
 Kohler, W. H., 279, 288, 300, 352
 Kohlstaedt, A. T., 170, 178, 352
 Kornfeld, W. A., 310–311, 315, 321, 352
 Kosaraju, S. R., 114, 129, 352
 label renaming, Concurrent Processes, 90
 lambda calculus, 8–12, 86–87, 90, 147, 149, 156, 336
 Lamport, L., 36, 40, 285, 300, 331, 343, 346, 352
 Lampson, B. W., 289, 300, 352
 Landin, P., 157, 164, 179, 352
 Lang, B., 228, 349
 languages, xiv
 pragmatic, 326
 Lauer, H. C., 49, 53, 353
 lazy evaluator, 159
 Ledgard, H., 16, 24, 353
 Lehmann, D. J., 141, 350
 Lesser, V. R., 302, 305, 317–318, 320–321, 349–350, 353
letrec, 162
 Levin, G. M., 130, 141, 144, 353
 Levin, M. I., 179, 353
 Lewis, P. M., 300, 354

- lexical process expansion, 48
 Li, C.-M., 200, 353
 Lieberman, H., 154, 351
 limited broadcast, Contract Nets, 307
 Lindsay, B. G., 288, 299, 347, 351
 Lindstrom, G., 173, 179, 352
 link, Data Flow, 115
 Liskov, B., 19, 24, 280, 290, 300, 353, 356
 Lisp, xv, xvii, 149, 157–158, 160, 165, 170
 exprs, 158
 pure, 17, 156, 336
 literal, 93
 Liu, M. T., 200, 353
 livelock, 33
 in CSP, 143
 liveness, Petri Nets, 109
 local area network, 41
 local storage, Distributed Processes, 192
 lock, exclusive (write, update), 283
 shared (read), 284
 locks, 36, 283
 logical processor, xiii
 long haul network, 41
 Lorie, R. A., 299, 350–351
 Lovelace, Countess of (Ada Augusta), 202
 Loveland, D. W., 92, 103–104, 353
 Lynch, N. A., 57–58, 60–61, 64, 71–73, 90,
 340, 348, 353
- McCarthy, J., 17, 24, 81, 85, 157, 160,
 179, 353
 McCullough-Pitts neural nets, 15
 McGraw, J. R., 127, 129, 353
 McJones, P., 299
 MacQueen, D. B., 157, 159, 166, 178–179,
 352–353
 MacQueen's merge, 166
 manifest values, suspending `cons`, 160
 Marcotty, M., 16, 24, 353
 markings, Petri Nets, 107
 mathematics of composition, Concurrent
 Processes, 89–90
 Mayr, E. W., 114, 129, 353
 Mead, C., 41, 43, 353
 megacomputers, xii
 merge of streams, 165–166
 merge, MacQueen's, 166
 solution in CSP, 134
 Turner's, 166
 message transmission, asynchronous, 49
 synchronous, 49
 messages, 18
 messages as name-value pairs, PLITS, 231
 messages, Actors, 146
 PLITS, 230
 Metcalfe, R. M., 77, 85, 353
 Michaelson, S., 178, 350
 Michie, D., 320, 349
 Micro-Planner, 310
 microprocessors, xii
 Miller, R. E., 106, 129, 352
 Milne, G. J., 86–88, 90, 95, 104, 340, 353
 Milner, R., 86–88, 90, 104, 178, 340,
 350, 353
 Minsky, M., 15, 354
 models, xiv
 Modula, 200
 Modula-2, 200
 module and message statements, PLITS,
 231–233
 modules, Ada, 204
 PLITS, 230
 Mohan, C., 325, 346, 354
 monitor node, Contract Nets, 308
 monitors, 36, 329
 Concurrent Pascal, 184
`continue`, 185
`delay`, 185
 initial operation, 185
 procedure entry, 185
 monotonicity, Scientific Community
 Metaphor, 310
 Moore, R. C., 316, 322, 354
 Morris, J. H., 159, 178, 351
 motivations, distributed databases, 280
 economic, xv
 intellectual, xv
 Muchnick, S. S., 244
 multiprocessing languages, xv
 multiprocessors, xiii
 mutual exclusion problem, 27–28
 mutual exclusion, solution in Petri Nets,
 110–111
 solution in Shared Variables, 62–63
 Myhrhaug, B., 23, 348
 Mylopoulos, J. L., 321, 351
- name, communication connection, 51
 Distributed Processes, 192
 names, guardians, 293
 Needham, R. M., 49, 53, 353
 negotiation-based control, heuristic
 mechanisms, 52
 Nelson, B. J., 49, 53, 354

- nested topaction, 293
- net, Concurrent Processes, 88
- network, local area, 41
 - long haul, 41
- networks of parallel processes, 159
- new**, Ada, 212
- Newell, A., 41, 43, 355
- Nii, H. P., 304, 322, 354
- noisy channel, 67
- normal termination, SR, 253
- normal-order evaluation, 158
- Nygaard, K., 23, 348

- object-oriented programming, 19, 145
- Omega, 149–150
- Open Systems, 315
- operating systems, xii, 326
- operating systems accountant, solution in Cell, 262–263
- operation command, SR, 248
- operation of barbershop, Petri Nets, 108
- operators, Data Flow, 116
- Organick, E. I., 16, 24, 354
- organization design language (ODL), 321
- organizational requirements, xviii
- output guards, CSP, 215
 - lack of in CSP, 143
 - solution in CSP, 135
- output statements, CSP, 131

- packages, Ada, 224
- parallelism, Scientific Community Metaphor, 310
- parbegin**, 21
- parend**, 21
- Parnas, D. L., 36, 40, 349
- partial order in **select** statement, Cell, 267
- partial-order priority, Cell, 273
- Pascal, xv, xvii, 202, 261, 336
- pattern-directed invocation, heuristic mechanisms, 52
- pattern-matched invocation, Actors, 149
- pending**, PLITS, 233
- peripheral devices, Concurrent Pascal, 187
- Perlis, A. J., 224, 228, 354
- Perlois, B., 128, 348
- permutation of events, 22
- Perrott, R. H., 40, 349
- Peterossi, A., 176, 179, 354
- Peterson, G. L., 62, 72–73, 348, 354
- Peterson, J. L., 34, 106–107, 110, 127, 129, 354

- Petri Nets, 106–114, 325–345
 - conditions, 110
 - events, 110
 - firing of transitions, 109
 - halting problem, 114
 - liveness, 109
 - markings, 107
 - operation of barbershop, 108
 - places, 106
 - postconditions, 110
 - preconditions, 110
 - reachability, 109
 - reachability problem, 114
 - readers-writers, 113
 - tokens, 107
 - transitions, 106
 - Turing-equivalence, 114
- Petri, C. A., 105, 129, 354
- places, Petri Nets, 106
- Planner, 310
- Plasma, 149
- PLITS, xviii, 230–245, 325–345
 - asynchronous communication, 230
 - messages, 230
 - messages as name-value pairs, 231
 - module and message statements, 231–233
 - modules, 230
 - pending**, 233
 - processes, 230
 - public slot names, 231
 - security in messages, 231
 - self destruct**, 233
 - slot, 231
 - transaction, 232
- Plummer, R. P., 16, 24, 354
- pluralism, Scientific Community Metaphor, 310
- Pnueli, A., 141, 350
- point-to-point announcement, Contract Nets, 307
- port, communication connection, 51
- ports, in Concurrent Processes, 87
- postconditions, Petri Nets, 110
- powerdomains, 86
- pragmatic languages, 326
- Pratt, T. W., 16, 24, 354
- preconditions, Petri Nets, 110
- Price, T., 299
- prime Fibonacci numbers, solution in Concurrent Pascal, 184–188
- primes, solution in IAP, 162–163

- primitive actor, 146
- priority clause, SR, 248
- priority mechanism, SR, 246
- problem domain, 46
- problem solving protocols, Contract Nets, 307
- procedure entry, monitors, 185
- process, xiii, 21
- process control, solution in Exchange Functions, 80–81
- process dynamics, dynamic creation, 48
 - lexical expansion, 48
 - static allocation, 48
- process scheduling, SR, 253
- process termination, Cell, 262, 268
 - SR, 249
- processes, 18
 - Ada, 202
 - Argus, 294
 - blocked, 27, 49
 - Cell, 261
 - Concurrent Pascal, 184
 - Contract Nets, 308
 - CSP, 131, 133
 - explicit, 48
 - in Concurrent Processes, 87
 - in Shared Variables, 57
 - PLITS, 230
 - SR, 246
 - synchronized, 27
- processing node, Contract Nets, 308
- processor, xiii
- producer-consumer buffer, 36
 - solution in Ada, 208–211
 - solution in Concurrent Pascal, 185–186
 - solution in CSP, 136–137
 - solution in Distributed Processes, 197
 - solution in Exchange Functions, 79–80
 - solution in IAP, 175
 - solution in PLITS, 235–237
 - solution in SR, 249–250
- production systems, 310, 320
- program, xiv
- program correctness, 326
- program, Data Flow, 115
- programming language, xiv
- programming languages, xv
 - semantics, xv
 - syntax, xv
- promotion algorithms, 167
- propositional calculus, 92
- protocols, solution in Shared Variables, 63–67
- public slot names, PLITS, 231
- pure Lisp, 17, 156, 336
- Putzolu, G. R., 299, 351
- Pyle, I. C., 202, 228, 354
- queue control, 337
- queue size, SR, 250
- raise** statement, Ada, 225
- Rao, R., 325, 346, 354
- reachability problem, Petri Nets, 114
- reachability, Petri Nets, 109
- read locks, Argus, 292
- read-write processes, in Shared Variables, 59
- readers-writers, 36
 - in Petri Nets, 113
 - solution in PLITS, 237–240
 - solution in SR, 250–253
- real time, 198
- real-time clock, in Exchange Functions, 76
- real-time control, 201
- real-time sensing, in Exchange Functions, 76
- recovery routine, guardians, 293
- recovery, Argus, 290
- recursion, data, 162
 - functional, 161
- Reddy, D. R., 320, 350
- redundancy, failure, 52
- Reed, D. P., 291, 300, 354
- register problem, 36
- register, solution in Cell, 261–262
 - solution in Concurrent Processes, 91
 - solution in Data Flow, 122
- Reid, L. G., 340, 346, 354
- remote procedure call, 49
- rendezvous, 329
 - Ada, 205
 - critical region, 205
- renewal, Concurrent Processes, 89
- replication transparency, 287
- replication, distributed databases, 279–280, 286–290
- request-response sequence, Contract Nets, 307
- requirements specification, 85, 326
- reset/set flip-flop, solution in IAP, 164–165
- resolution algorithm, 93
- resource, 27

- resource conflict, 25–27
- resources, SR, 246
- restricting label visibility, Concurrent Processes, 90
- restrictions, **select** alternatives, 207
- Reynolds, J. C., 144, 146, 154, 354
- Riseman, E. M., 321, 351
- Robinet, B., 128, 346, 349
- Robinson, J. A., 92, 104, 354
- Robson, D., 23, 350
- Rodriguez-Bezos, J. E., 106, 129, 354
- Roitblat, B., 356
- Roland, V., 356
- rollback, transaction, 283
- Rosenfeld, J. L., 178, 352
- Rosenkrantz, D. J., 286, 300, 354
- Rosser, J. B., 10
- run-time structure, xviii
- Ruschitzka, M., 178, 352
- Sacerdote, G. S., 114, 129, 354
- Saint, H., 129, 352
- Schaffert, J. C., 24, 300, 353
- schedule, definition of, 282
 - serial, 282
 - serializable, 282
- scheduler, Hearsay-II, 304
- scheduling, Cell, 263–265
- Scheifler, R., 300, 353
- Scheme, xviii, 149–150, 156, 158
- Schmidt, J. W., 321, 351
- Schneider, F. B., 325, 346–347
- Scientific Community Metaphor, 303, 309–317, 325–345
 - commutativity, 310
 - monotonicity, 310
 - parallelism, 310
 - pluralism, 310
- Scott, D. S., 15
- scripts, Actors, 146
- security, xv
- security in messages, PLITS, 231
- Seegmuller, G., 299, 351
- select** alternatives, **accept**, 206
 - delay**, 206
 - else**, 207
 - guarded, 206
 - restrictions, 207
 - terminate**, 206
- select** statement, Ada, 206
 - Cell, 264
- select/entry call, Ada, 214–215
- self destruct**, PLITS, 233
- semantics, 326
 - axiomatic, 18
 - denotational, 18
 - programming languages, xv
- semaphore, 35, 175
 - binary, 35
 - solution in Distributed Processes, 194
 - solution in Shared Variables, 61–62
- send-and-forget, 49
 - in IAP, 169
- sensing node, Contract Nets, 308
- serial schedule, 282
- Serializable schedule, 282
- serialized access, 27
- serialized actor, 146–147
 - bank account example, 152
- Shapiro, R., 129, 352
- shared (read), lock, 284
- shared memory, xiii
- Shared Variables, xviii, 57–72, 245, 325–345
 - advantages, 57
 - processes, 57
 - read-write processes, 59
 - solution in Exchange Functions, 78–79
 - test-and-set, 59
- Shaw, A. C., 40, 355
- Shaw, M., 19, 24, 355
- side effects, IAP, 298
- sieve of Eratosthenes, 162
- Siewiorek, D. C., 41, 43, 355
- Silberschatz, A., 70, 72, 142–144, 227–228, 246, 261, 264, 275, 348, 352, 355
- Simi, M., 154, 351
- simple-merge actors, Data Flow, 117
- simplex connection, 50
- Simula 67, 19
- Sinclair, J. B., 70, 72, 351
- slot, PLITS, 231
- Smalltalk, 19
- Smith, R. G., 307, 319–320, 322, 349, 355
- Smyth, M. B., 89, 104, 355
- Snyder, A., 24, 300, 353
- solution in Actors, bank account, 152–153
 - factorial, 151–152
- solution in Ada, dining philosophers, 212–214
 - elevator controller, 215–224
 - producer-consumer buffer, 208–211
- solution in Argus, dining philosophers, 294–298
- solution in Cell, Exchange Functions, 265

- operating systems accountant, 262–263
- register, 261–262
- solution in Concurrent Pascal, dining philosophers, 188–189
- prime Fibonacci numbers, 184–188
- producer-consumer buffer, 185–186
- solution in Concurrent Processes, binary semaphore, 92
- card reader, 95–103
- register, 91
- theorem proving, 92–95
- solution in CSP, dining philosophers, 137–138
- Fibonacci numbers, 133
- merge, 134
- output guards, 135
- producer-consumer buffer, 136–137
- sorting tree, 138–141
- solution in Data Flow, airline reservation system, 124–125
- conditionals, 117–119
- demultiplexer, 120–121
- exponentiation, 119–120
- register, 122
- solution in Distributed Processes, binary semaphore, 194
- dining philosophers, 194–197
- producer-consumer buffer, 197
- semaphore, 194
- solution in Ether, covering set problem, 311–316
- solution in Exchange Functions, binary semaphore, 78
- process control, 80–81
- producer-consumer buffer, 79–80
- Shared Variables, 78–79
- vat problem, 80–81
- solution in IAP, algorithmic selection, 171–172
- cull, 174–176
- good sequences, 170
- Hamming sequence, 163
- primes, 162–163
- producer-consumer buffer, 175
- reset/set flip-flop, 164–165
- successors, 162
- terminal controller, 172–173
- solution in Petri Nets, dining philosophers, 111–112
- mutual exclusion, 110–111
- solution in PLITS, eight queens, 240–243
- Fibonacci numbers, 233–235
- producer-consumer buffer, 235–237
- readers-writers, 237–240
- solution in Shared Variables, communication delay, 67–68
- continuous display, 70–71
- election algorithms, 68–70
- mutual exclusion, 62–63
- protocols, 63–67
- semaphore, 61–62
- solution in SR, producer-consumer buffer, 249–250
- readers-writers, 250–253
- traffic light control system, 254–261
- sort, Concurrent Processes, 88
- sorting tree, solution in CSP, 138–141
- space complexity, 13
- speech understanding, Hearsay-II, 304
- split, 336
- split**, filling, 337
- split, in IAP, 174
- sprites, Ether, 310
- SR, 245–261, 268–273, 325–345
- abnormal termination, 253
- asynchronous sends, 246
- by**, 248
- guarded commands, 248
- in** statement, 248
- normal termination, 253
- operation command, 248
- priority clause, 248
- priority mechanism, 246
- process scheduling, 253
- process termination, 249
- processes, 246
- queue size, 250
- resources, 246
- synchronous calls, 246
- stable storage, Argus, 290
- guardians, 293
- Stark, E. W., 40, 355
- starvation, 27
- statements, CSP, 132
- static process allocation, 48
- Stearns, R. E., 300, 354
- Steele, G. L., xviii, 19, 24, 146, 149–150, 154–155, 179, 355
- sting, in IAP, 169
- storage allocation, Concurrent Pascal, 188
- Stotts, P. D., 325, 346, 355
- Stoy, J. E., 15, 18, 24, 355
- Strachey, C., 15, 146, 155, 355
- streams, 157

- strong fairness, 51, 333
 Sturgis, H. E., 300, 352
 subactions, Argus, 291
 successor functions, Exchange Functions, 74, 77
 successors, solution in IAP, 162
 suspending **cons**, 159
 forcing, 160
 manifest values, 160
 suspending, **cons**, 159–161
 Sussman, G. J., 24, 149, 155, 179, 310, 321–322, 355
 Sutherland, I., 106
 symmetric communication control, 50
 synchronization, xv
 synchronization mechanisms, 28–36
 synchronized communication, Ada, 202, 226
Synchronizing Resources, *see SR*
 synchronous calls, SR, 246
 synchronous communication, 327
 synchronous message transmission, 49
 syntax, programming languages, xv
 Syre, J.-C., 106, 128, 348
 system analysis, xiv
 system design, xiv
 system validation, xiv
 systems implementation, 326
- Tanenbaum, A. S., 42–43, 355
 target Actors, 146
 task description, Contract Nets, 307
 task priority, Ada, 224–225
 task report, Contract Nets, 307
 task termination, Ada, 203
 task types, Ada, 212
 tasks, Ada, 203, 232
 tautology, 92
 Tenney, R. L., 129, 354
 terminal controller, solution in IAP, 172–173
terminate, select alternatives, 206
 termination, Ada, 225–226
 Tesler, L. G., 106, 129, 355
 test-and-set, 34
 in IAP, 170
 in Shared Variables, 59
 theorem proving, solution in Concurrent Processes, 92–95
 thunks, 19, 149
 time complexity, 13
 time-out, 51
 time-outs, failure, 52, 333
 timed entry call, Ada, 215
 timesharing systems, xii–xiii
 timestamp, 286
 timestamps, 200, 250, 283, 285
 tokens, Petri Nets, 107
 topactions, Argus, 291
 traffic light control system, solution in SR, 254–261
 Traiger, I. L., 299, 350–351
 transaction, die, 286
 PLITS, 232
 rollback, 283
 two-phase, 283
 wait-die, 286
 well-formed, 283
 wound, 286
 wound-wait, 286
 transactions, distributed databases, 280–286
 transitions, Petri Nets, 106
 transparency, distributed databases, 280
 Treleaven, P. C., 105, 127, 129, 355
 triple machine, 4
 Turing machines, 6
 Turing, A., 6–7, 15, 356
 Turing-equivalence, Petri Nets, 114
 Turing-equivalent automata, 7
 Turner, D. A., 166, 174, 179, 356
 Turner's merge, 166
 two-phase commit, 285, 288–290
 Argus, 290
 two-phase, transaction, 283
- Ullman, J. D., 15, 279, 300, 347, 352, 356
 unbounded buffers, 50, 329
 unidirectional information flow, 50, 329
 unserialized actor, 146
- Van Horn, E. C., 40, 349
 vat problem, solution in Exchange Functions, 80–81
 versions, Argus, 292
 Vichnevetsky, R., 178, 352
 virtual circuits, 50
 VLSI, 41
 volatile storage, Argus, 290
 guardians, 293
 von Neumann, J., 3, 17, 105, 154
 Vuillemin, J., 159, 179, 356
- Wadge, W. W., 177, 348
 Wadsworth, C. P., 155, 159, 179, 355–356
 wait-die, transaction, 286
 waiting-for relation, deadlock detection, 285
 Wand, M., 18, 24, 159, 179, 356

- Warshall, S., 129, 352
Waterman, D. A., 322, 354
Watson, I., 106, 129, 356
weak fairness, 51, 333
Wegner, P., 15, 202, 229, 356
Weihl, W., 300, 356
well-formed, transaction, 283
Weyhrauch, R. W., 316, 322, 356
when statement, Ada, 206
when, Distributed Processes guarded commands, 193
Wiederhold, G., 279, 300, 356
Williams, J. H., 128, 349
Williams, J.W.J., 138, 144, 356
Winograd, T., 316, 322, 355–356
Winston, P. H., 160, 179, 356
Wirth, N., 178, 200, 244, 356
Wise, D. S., 159–160, 164, 166, 168–169, 177–178, 350
Wolynes, G. P., xviii, 356
wound, transaction, 286
wound-wait, transaction, 286
write locks, Argus, 292
X, Exchange Functions, 75
X.25 protocol, 42
XM, Exchange Functions, 76
XR, Exchange Functions, 75
Y combinator, 11
Zave, P., 74–77, 84–85, 350, 356